

Improvement of Random Forest Classifier through Localization of Persian Handwritten OCR

M. Zahedi, S. Eslami

Shahrood University of Technology, Shahrood, Iran

{Zahedi, Saeideh_eslami} @shahroodut.ac.ir

Abstract — The random forest (RF) classifier is an ensemble classifier derived from decision tree idea. However the parallel operations of several classifiers along with use of randomness in sample and feature selection has made the random forest a very strong classifier with accuracy rates comparable to most of currently used classifiers. Although, the use of random forest on handwritten digits has been considered before, in this paper RF is applied in recognizing Persian handwritten characters. Trying to improve the recognition rate, we suggest converting the structure of decision trees from a binary tree to a multi branch tree. The improvement gained this way proves the applicability of the idea.

Index Terms— Random forest classifier, Persian/ Arabic alphabet, handwritten characters, loci features, splitting function

I. INTRODUCTION

The problem of character recognition OCR has widely been the object of interest in the machine learning and AI issues. Handwritten character recognition entails its own challenges as careless and rough handwritings often misguide the classifier. Cursive alphabets like Arabic and Persian add to the complexity of the task hence the number of relevant researches is rather few [1]. In this paper, random forest classifier is used to recognize 33 Persian handwritten single characters. The random forest (RF) is a robust and fast classifier which is also able to handle large number of feature variables. Also it is suggested to improve the efficiency of the RF through a localization process which makes the structure of decision trees like a multi-class tree. The implementation results verify the theory. The characteristics of Persian/Arabic alphabet will be discussed in detail in the next part. We go over a brief review on the common features and methods used for Farsi/Arabic handwritten OCR in section III. Section IV summarizes the loci features which are used in recognition. A synopsis of random forest structure and its classification strategy comes in section V. Part VI and VII discusses in turn some main evolution algorithms applied on random forest algorithm and our localization theory. Finally the section VIII gives the experimental results and diagrams.

II. CHALLENGES IN PERSIAN/ARABIC CHARACTER RECOGNITION

Most of the difficulty in Persian/ Arabic OCR goes back to the large number of different letters in these languages. The presence of dotted characters causes another challenge. 16 out of 32 Persian characters contain dots. Some of these characters can be grouped based on their number of dots. It means the only difference within such a group is the number

or location of the dots while the basic shape of these letters is totally the same. The next table presents printed Persian characters grouped based on their basic shape [2]. Besides; most of characters may appear both in isolated and cursive form; and there are several cursive forms for some of

TABLE I.
THE DOT-LESS FORM OF CHARACTERS IN EACH COLUMN IS THE SAME

Group								
#1	#2	#3	#4	#5	#6	#7	#8	#9
Baa	Haa	Daal	Ra	Siin	Saad	Taa	Ain	Faa
→	ح	د	ر	س	ص	ط	ع	ف
→	خ	ذ	ز	ش	ض	ظ	غ	ق
→	ج		ژ					
→	چ							
→	گ							

characters, depending on their location in a word [3]. These cursive forms make a total number of more than 56 characters along with their single forms.

TABLE II.
SOME CHARACTERS HAVE SEVERAL CURSIVE FORMS DEPENDING ON THEIR POSITION IN THE WORD.

Group	Different Forms of Characters	
	Isolated form	Dot-less Cursive form(s)
#1	ح ح ح ح	ح ح ح ح
#2	س ش	س ش س ش
#3	ص ض	ص ض ص ض
#4	ع غ	ع غ ع غ
#5	ک گ	ک ک ک ک
#6	ه	ه ه ه ه
#7	م	م م م م
#8	ب پ ت ث	ب ب ب ب

The problem is worse when it comes to handwritten documents. The form of writing dots in 2 or 3-dotted characters vary for different writers. The next figure illustrates some common forms of writing dots.

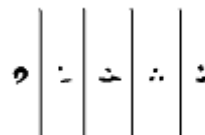


Fig. 1. Different forms of writing 3-dot components

As a result of careless writing, dots can be misplaced or can have an unreasonable distance from the character they belong to. This can lead to a notable decrease in the recognition accuracy while noise and image degradation make the problem much complicated.

II. CLASSIFICATION METHODS AND FEATURES

Characters can be classified using structural, statistical and global features. Structural features refer to the skeleton or basic topology of the character such as loops, branch-points, zigzags, height, width, number of crossing points and endpoints. They are also very helpful in extracting dot information to classify letters [4], [5]. Statistical features are numerical information like pixel densities, histogram of chain code directions, moments, characteristic loci and Fourier descriptors [4], [5], [6]. Statistical features are effective and fast to compute but may be affected by noise [5]. Global transformations methods mostly include: horizontal and vertical projections, coding, Hough transform, Gabor transform [5]. Hidden Markov model is known to learn and find characteristics that are difficult to describe [4]. So; structural features can be used very well with HMM to recognize a letter. Some researchers suggest using a combination of classifiers like HMM with SVM, SOM, RBF or another neural network [7], [8], [9]. Classification results of such systems are promising but the high complexity and required training time are considered as their drawbacks.

III. FEATURE EXTRACTION

The loci features have been proven to be very prosperous in OCR [10]. The notable increase in the recognition accuracy has emerged many researchers to use them in handwritten OCR as well [10], [11]. The loci features for each sample image build an 81-length feature vector. Taking only background pixels into account, the number of times a direct line starting from a center pixel and continuing to reach the image boundary, passes through a foreground area in each north, south, east and west directions, is calculated and converted to a ternary code according to the following formula:

$$\begin{aligned} \text{value} = & \text{east_counter} * 27 + \\ & \text{west_counter} * 9 + \text{north_counter} * 3 \\ & + \text{south_counter} \end{aligned} \quad (1)$$

where “east_counter” is the obtained counter during eastward projection and similarly for other counters. Note that there is no priority between counters and they can be replaced but a fixed format should be used for all cases. The feature vector is not affected by the size of image. Using ternary code, there will not be more than 81 different values per image. The 81-length histogram of the counters will build up the feature vector of the image. This is fixed for all images of all sizes.

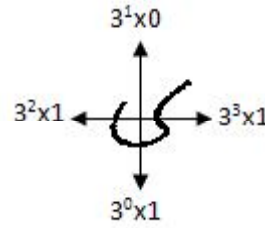


Fig. 2. Projection in 4-main directions to compute loci features

IV. RANDOM FOREST CLASSIFIER

Boosting is an ensemble classification method when each individual classifier try to contribute to the final result by improving the result received from its previous counterpart. Bagging also means building an ensemble of base classifiers, each one trained on a bootstrap replicate of the training set. Decisions are then made on the basis of voting [12]. Based on the bagging theory [13], Breiman introduced the concept of random forests in 2001 [14]. A decision forest is an ensemble of several decision trees which act in a parallel manner. The decision forest is different from boosting in way that each tree tries to classify the data totally independent of the other trees [13]. Breiman suggested that randomness can be applied on the selection of training data samples and set of features used to classify the data [14]. The reason is not all features contribute to the correct recognition and some may deteriorate the results. The accuracy of RF depends on:

- The correlation between each pair of trees in the forest; Increasing the correlation or dependency of the trees in prediction, increases the error rate as well.
- The strength of each single tree in the forest; Increasing the strength of the individual trees increases the forest's overall recognition accuracy [15].

With enough number of classifiers (decision trees) and different sets of features, we can hope that only the most persistent features exist in the data for classification and the irrelevant features do not affect its efficiency so much [16]. The random selection also helps the classifier to handle missing values. If the number of trees is controlled, the random forest classifier can escape overfitting too.

A. Random forest training

All the data is fed to all trees at once. For each tree, a set of 'n' samples are randomly extracted with replacement from the available training set. Again at each node, a random set of features consisting of a fixed number of feature variables (often equal to the square root of the total number of features) is selected from the feature vector [17]. Several random functions and linear combinations are produced out of selected features and are tested to classify the data. Training means finding the best splitting formula or the best linear combination of feature variables. Such a formula minimizes a cost function or rather maximizes an attribute evaluation function and splits the data in the best way at that node [14]. Random forests use the Gini criterion derived from the CART

decision trees [18], [19].

$$Gini = \sum_{i=0,1} p_i (1 - p_i) \quad (1)$$

The other important criterion is the entropy function which is calculated as follows:

$$Entropy = \sum_{i=0,1,2,\dots,n} -p_i \log(p_i) \quad (2)$$

where, p_i stands for proportions of data samples from class i . At each node of each tree, a set of random functions and linear combinations are produced [16]. The best function is selected to classify the future samples at that node. The nodes often break until there is only one sample in each node. The probability of belonging to a class for each data sample is calculated. The same process runs through other trees. The final class is determined by the voting mechanism; i.e. the class chosen by most of trees for a data sample is declared as its final class.

B. Random forest parameters

The determinant factors in RF classifier's recognition rate are the number and depth of the decision trees, number of splitting features and the minimum number of remained samples to need a node split into another level. The most critical parameter which needs to be adjusted is the number of randomly selected variables used to form a splitting criterion at each node. The splitting formula divides the available data into different categories. However the recognition accuracy is not very sensitive to chosen value as long as its difference with the real optimum value is eligible. So, based on simple assumptions, the default value is often set to the square root of number of features which gives rather near optimal results and can also be used as a start point in searching for the real optimum value [16], [17].

V. EVOLUTION ALGORITHMS

Since the introduction of RF, there has been no standard study on adjustment of critical parameters such as number of trees, number of used features and depth of a decision tree. The set of training parameters are often adjusted according to some default formulas which produce acceptable results but they have been gained by try and error. As it is mentioned before, the number of features used in splitting is the most crucial variable since as the number of used features increases, the possibility of using similar sets of features also increases. The consequent correlation between different trees has proved to reduce the recognition rate. A near optimal value is the square root of the number of features used as the default number of splitting features but without any verification. The number of trees commonly varies between 100 and 150. Greater values do not improve the recognition much and there is the possibility of overfitting for small datasets. In contrast to parameter evaluation, some researches did suggest some innovative methods to increase the overall efficiency of the random forest by moderate modifications on the original algorithm [15]. Breiman first proposed to use

a single random feature to split the data at each node [13] but he later suggested using a linear combination of several features instead of one [14], [17]. The latter results in less correlation between trees and better management of missing feature values. The accuracy of recognition can be increased by eliminating or lowering the effect of weak classifiers, i.e. the trees with high rate of misclassification. Robnik's weighted voting system selects a subset of decision trees as the classifiers using the accuracy records gained during training phase [12]. The system finds the training sample(s) with most similarity to the unknown sample. The trees which have mislabeled such cases are then put away to avoid further error. Finding similar samples follows the algorithm in [17]. It is claimed that weighted voting is often better and never worse than the original voting method. This method acts similar to the AdaBoost algorithm in which the training set of each classifier is weighted according to the correctness of the prediction [12]. The main reason of the random forest's strength and efficiency is the induction of randomness both in sample and feature selection. It is the Gini criterion or the entropy function that decides on the selection of splitting threshold but in [18], [20] it is reported that a totally random cut-point also yields comparably accurate results. Although the individual tree classifiers are very weak, they are uncorrelated [20]. Boine et al. [18] tries to build a better classifier by combining the AdaBoost and Bagging techniques with the random forests. The paper actually proposes to use an ensemble of random forests instead of the original random forest algorithm based on the theory that a combined classifier often acts better than individual classifiers.

VI. LOCALIZATION

There is no limit in the number of categories when using RF however the random forest actually builds up a binary structure. This means the inputs break into two branches when reaching a node. They go down the tree until there is no sample remains unclassified. In each non-leaf node, a unique splitting function is selected among other randomly produced ones when the function splits the data better than the others. The problem of Persian character recognition has been discussed before in the paper and it is noted that some characters appear basically in the same group ignoring their dots. Complication occurs when trying to distinguish such characters in their own group. Here we suggest applying a separate splitting function for each group which recognizes these characters inside the group. Thus each node converts to a multi-branch classifier rather than the previous binary one. In our case the decision tree becomes a 10-branch structure with single characters being recognized all with one common splitting function. The improvement gained in recognition is depicted in next section.

VII. EXPERIMENTAL RESULTS

The training database consists of 2838 binary image samples of 33 Persian single characters written by 86 different

writers. A complete set of 1650 samples was used to train the classifier and the rest 1180 were used as the test set to verify its efficiency. The inputs to classifier were loci feature vectors of length 81 extracted from binary images. As there is no verified formula to find optimal parameter values for the forest, numerous trials were done with different adjustment of RF parameters (number of trees and splitters). The diagrams depict the effect of changing the training variables on recognition rate and also on the elapsed training time. In the initial phase the random forest is tested using its original structure. The highest accuracy was reached using 120 trees and 7 random splitting features with 87% of the samples recognized correctly. The total time needed for training and testing a 120-tree forest with 2838 feature vectors of size 81, is just about 5s. More detailed results are shown in fig. 3. Part of error goes back to those dotted groups mentioned earlier in the paper. Trying to fix the error, these characters can be grouped and considered as one class. Another set of features then can be utilized to classify the inter-group characters. Results of such classification are compared to the previous ones in fig. 4. The next two figures demonstrate the localization effect on the recognition accuracy. There is an apparent improvement on the recognition both in the original classification and in the joint-class character recognition problem. The implementation of the localization idea raises the accuracy rate to 94%. The average accuracy for Persian handwritten OCR reported by different researchers on different datasets is currently about 95% [21], [22], [23]. Nevertheless the main focus of this paper is not improvement of Persian handwritten OCR but proposing a method for increasing the random forest's recognition rate especially when the similarity between different classes becomes very troublesome.

VIII. CONCLUSION AND FUTURE WORK

In this paper, the use of random forest in the field of Persian handwritten character recognition has been discussed. Most of Persian/Arabic characters can be grouped based on their basic shape. The main challenge in Persian OCR goes back to distinguishing such characters. A similar problem occurs when samples belonging to different classes have several common features. In such cases a single entropy function may not recognize the difference between the samples. Thus we suggested using several entropy functions each dedicated to one of these groups rather than one global entropy criterion. The improvement in recognition accuracy shows possibility of further improvement in recognition by random forests. The coding also has great impact on the results. Professional coding and better structures can increase both speed and accuracy. We are going to test the contribution on other datasets and features to evaluate its applicability in different areas.

REFERENCES

- [1] S. Mozaffari and H. Soltanizadeh, "ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition", 10th International Conference on Document Analysis and Recognition, 2009.
- [2] H. Izakian, S. A. Monadjemi, B. Tork Ladani, and K. Zamanifar, "Multi-font Farsi/Arabic isolated character recognition using chain codes", *World Academy of Science, Engineering and Technology*, vol. 43, 2008.
- [3] M. Zahedi, S. Eslami, "Farsi/Arabic Optical Font Recognition Using SIFT Features", *World Conference on Information Technology (WCIT)*, *Procedia Computer Science*, vol. 3, pp. 1055-1059, 2011.
- [4] L.M. Lorigo, V. Govindaraju, "Off-line Arabic Handwriting Recognition: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 712-724, 2006.
- [5] A. M. AL-Shatnawi, S. AL-Salaimeh, F. H. AL-Zawaideh and K. Omar, "Offline Arabic Text Recognition – An Overview", *World of Computer Science and Information Technology Journal (WCSIT)*, Vol. 1, No. 5, 184-192, 2011.
- [6] R. J. Ramteke, "Invariant Moments Based Feature Extraction for Handwritten Devanagari Vowels Recognition", *2010 International Journal of Computer Applications*, vol. 1, No. 18.
- [7] M. Dehghan, K. Faez, M. Ahmadi and M. Shridhar, "Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM", *Pattern Recognition* vol. 34, pp. 1057-1065, 2001.
- [8] N. Ben Amor, "Multifont Arabic Characters Recognition Using Hough Transform and HMM/ANN Classification", *Journal of Multimedia* vol. 1, no. 2, 2006.
- [9] R. Al-Jawfi, "Handwriting Arabic Character Recognition LeNet Using Neural Network", *The International Arab Journal of Information Technology*, vol. 6, no. 3, 2009.
- [10] I.D. Trier and A.K. Jain, "Feature extraction Methods for Character recognition- A Survey", *Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.
- [11] R. Ebrahimpour, M.R. Moradian, A. Esmkhani, F. M. Jafarlou, "Recognition of Persian handwritten digits using Characterization Loci and Mixture of Experts", *International Journal of Digital Content Technology and Its Applications*, vol. 3, no. 3, pp. 42-46, 2009.
- [12] M. Robnik, "Improving Random Forests", *Machine Learning*, ECML, Springer, Berlin, 2004.
- [13] Breiman, L., "Bagging predictors", *Mach. Learn.*, vol. 24, pp. 123-140, 1996.
- [14] Breiman, L., "Random forests", *Mach. Learn.*, vol. 45, pp. 5-32, 2001.
- [15] S. Bernard, L. Heutte, S. Adam, "Using Random Forests for Handwritten Digit Recognition", *Proceedings of the 9th IAPR/IEEE International Conference on Document Analysis and Recognition ICDAR'07*, Curitiba : Brazil, 2007.
- [16] D. Amaratunga, J. Cabrera and Y. S. Lee, "Enriched random forests", *Bioinformatics*, Vol. 24, No. 18, pp. 2010-2014, 2008.
- [17] Breiman, L. and Cutler, A., "Random forests manual (version 4.0). Technical Report of the University of California, Berkeley, Department of Statistics, 2003.
- [18] P. Boinee, A. De Angelis and G. L. Foresti, "Meta Random Forests", *International Journal of Information and Mathematical Sciences*, 2006.
- [19] R. B.H. Menza, B.M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich and F.A. Hamprecht, "A Comparison of Random Forest And Its Gini Importance With Standard Chemometric Methods For The Feature Selection And Classification of Spectral Data", *BMC*, 2009.
- [20] A. Cutler and G. Zhao, "PERT- Perfect Random Tree Ensembles", *Computing Science and Statistics*, vol. 33, 2001.
- [21] A. Ebrahimi and E. Kabir, "A pictorial Dictionary for Printed Farsi Subwords", *Pattern Recognition Letters* vol. 29, No. 5, 656-

663, 2008

- [22] Borji A, Hamidi M, Mahmoudi F (2008) Robust Handwritten Character Recognition With Features Inspired By Visual Ventral Stream. Neural Process Letters vol. 28, No. 2, 97–111.
- [23] M. Dehghan and K. Faez, “Farsi handwritten character recognition with moment invariants”, Proceedings of 13th International Conference on Digital Signal Processing-DSP-97, vol. 2, 1997.

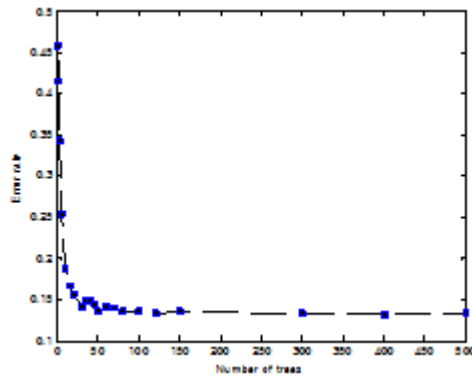


Fig. 3. RF classification results, with different number of decision trees in the forest.

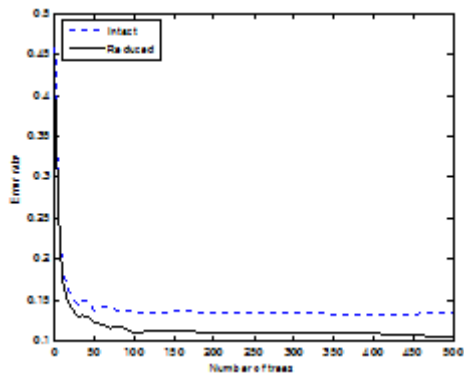


Fig. 4. Comparative diagram: the results before and after joining similar classes

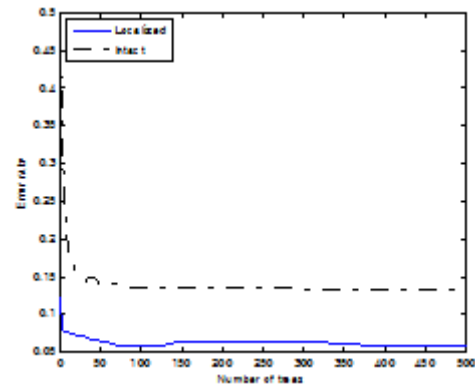


Fig. 5. Comparative diagram: the results before and after localization; localization error versus original state;

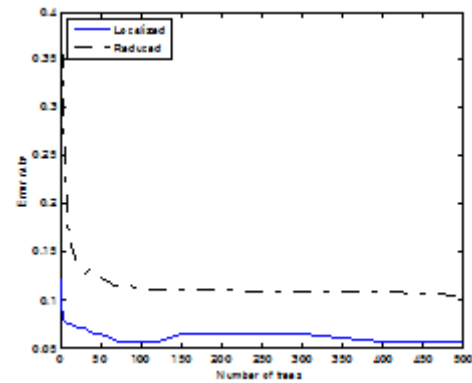


Fig. 7. Comparative diagram: the results before and after localization; Localization error versus joint class state.